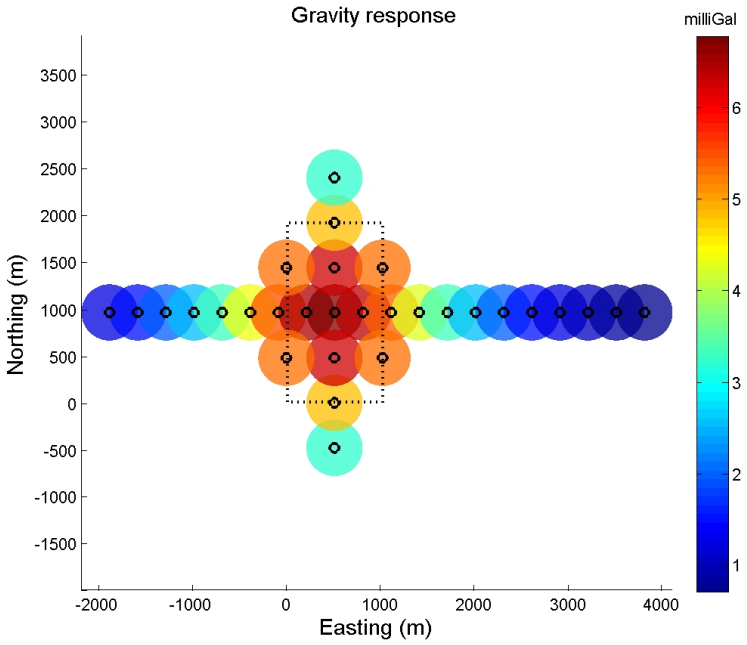
Norsk Regnesentral
NORWEGIAN COMPUTING CENTER

Note

Inversion of gravimetric data



Note no SAND/08/13

Authors Vera Louise Hauge
Odd Kolbjørnsen

Date June 13, 2013

Norwegian Computing Center

Norsk Regnesentral (Norwegian Computing Center, NR) is a private, independent, non-profit foundation established in 1952. NR carries out contract research and development projects in information and communication technology and applied statistical-mathematical modelling. The clients include a broad range of industrial, commercial and public service organisations in the national as well as the international market. Our scientific and technical capabilities are further developed in co-operation with The Research Council of Norway and key customers. The results of our projects may take the form of reports, software, prototypes, and short courses. A proof of the confidence and appreciation our clients have in us is given by the fact that most of our new contracts are signed with previous customers.

Norsk Regnesentral
Norwegian Computing Center
Postboks 114, Blindern
NO-0314 Oslo, Norway

Besøksadresse
Office address
Gaustadalléen 23
NO-0373 Oslo, Norway

Telefon · telephone
(+47) 22 85 25 00
Telefaks · telefax
(+47) 22 69 76 60

Internett · internet
www.nr.no
E-post · e-mail
nr@nr.no

Title	**Inversion of gravimetric data**
Authors	**Vera Louise Hauge** <vera.louise.hauge@nr.no> **Odd Kolbjørnsen** <odd.kolbjornsen@nr.no>
Date	June 13, 2013
Publication number	SAND/08/13

Abstract

This note is written in the project "Monitoring Geological CO_2 Storage: Quantitative CO_2 Predictions with Uncertainty from Physical Modelling and Multiple Time-Lapse Data Types" and describes the geostatistical inversion of gravimetric data. To overcome computational complexity, we apply upscaling to reduce the number of unknown parameters in the inversion. The note describes in detail the upscaling, geostatistical inversion as well as the back transform of the estimates to the full size domain. The methodology is exemplified by an example using synthetic data. The results show that gravimetric data carry information about low-frequent components in the density change. We see that gravimetric inversion alone is not sufficient for infering local details about the density. However, gravimetric data can verify properties on a larger scale, such as average density.

Keywords	Geophysical inversion, gravimetry
Target group	Project group
Availability	Internal
Project	Monitoring Geological CO2 Storage
Project number	808007
Research field	Geophysical inversion
Number of pages	36

Contents

1 Introduction

This note will describe the Bayesian inversion of gravimetric data. Gravimetric measurements give constraints on subsurface density variations. For the purpose of learning we first describe in simple terms the contributions to the gravimetric measurements. Next, we relate gravimetric data with subsurface parameters in a Gaussian-linear framework and find expressions for the conditional inverse problem. Due to large memory and computational costs of the inverse problem, the need for upscaling appears. Thus upscaling by use of convolution and subsampling in the Fourier domain is described. Most descriptions are accompanied by illustratory MATLAB examples. Finally, a MATLAB example with synthetic data and gravity inversion is included.

2 Setting the scene

The Sleipner Project has introduced use of gravimetric data to constrain density of injected CO_2 as a complement to the use of seismic data. Generally, 4D seismic surveys give a good image of the subsurface. Particularly in the Sleipner Project it gives a good image of the geometry of the CO_2 plume. However, there are large uncertainties related to the density of the CO_2 in the subsurface which the seismic surveys do not resolve. Gravimetric observations over a time span detect changes in density and are thus used as a complement to the seismic surveys. Gravimetric observations are measured at different times, here referred to as vintages, and the changes observed at different vintages are assigned to changes in mass of injected gas. Thus the relative changes in gravimetric observations hold the information used to constrain the mass and density.

Published literature using gravimetric data to estimate large scale properties of injected CO_2 include (Nooner et al., 2007), (Alnes et al., 2008), and (Alnes et al., 2011).

In the seismic setting vertical depth below sea bed is measured in terms of two-way-travel time of seismic waves and the velocity of the P-wave. This convention will be used in the following.

The 4D inversion scheme consists of 3D AVO (amplitude) inversion of seismic data that has been extended to include time (4D) by use of rock physics relations to evolve seismic quantities in time (Kolbjørnsen and Kjønsberg, 2011). Seismic parameters solved for are velocities for P-wave and S-wave and rock density. Inversion of two more data sets, two-way-travel time and gravity, are added in the time sequence.

Working with seismic inversion the unknown variables that are solved for are probability distribution functions. More particularly, the seismic parameters are log-normally distributed. However, in the following of this note we write all expressions and calculations with the assumption that the seismic parameter itself is normally distributed. See Appendix A.

3 Gravimetric observations

A gravimeter is the instrument used to measure gravity. The term gravimetry denotes the measurement of the strength of the gravitational field. In this note we will consider gravimetric measurements and relate it to the subsurface mass.

We first consider a gravimeter located at the seabed above a subsurface reservoir. Each point in the earth, thus both in the reservoir and surrounding areas, contributes to the measurements in the

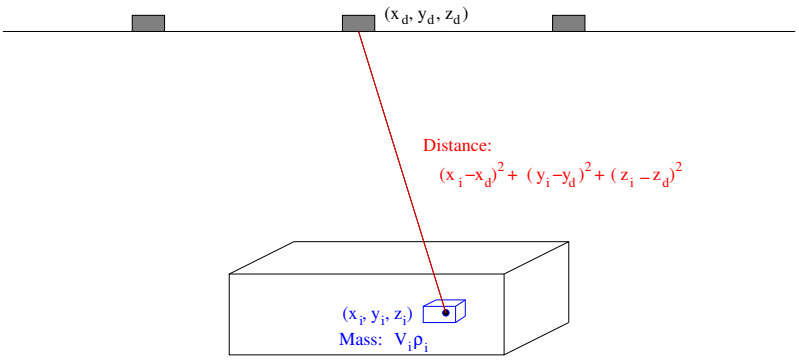


Figure 1. Sketch of reservoir, sea bed and gravimetric observations points.

gravimeter, henceforth referred to as gravimetric observations. The contribution from each point is proportional to m/r^2 , where m is the mass of the point and r is the distance from the point in the subsurface reservoir to the gravimeter located at the seabed. (Newton's law of gravitation).

Applied to a reservoir model consisting of a regular grid, where each grid cell has dimensions $\mathrm{d}x \times \mathrm{d}y \times \mathrm{d}z$, the grid cell representing the position (x_i, y_i, z_i) has a contribution to the gravimeter located at position (x_0, y_0, z_0):

$$\gamma \frac{m_i \rho_i}{r_{i,0}^2} = \gamma \frac{(\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z)\rho_i}{r_{i,0}^2} = \gamma \frac{\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}t \frac{v_p}{2} \rho_i}{r_{i,0}^2} \tag{1}$$

where γ is a gravitational constant, ρ_i is the density, $\mathrm{d}t$ is the two-way-travel time of the seismic wave, v_p is the P-wave seismic velocity, and $r_{i,0}^2 = ||(x_i, y_i, z_i) - (x_0, y_0, z_0)||^2$. The vertical length of a grid cell $\mathrm{d}z$ is expressed in terms of two-way-travel time and seismic P-wave velocity, $\frac{v_p}{2}\mathrm{d}t$. When relating this contribution from the reservoir and observed data we consider both the density and the P-wave velocity as unknowns. Thus the unknown variable is $v_p\rho$ and will be referred to as seismic parameter in the following.

The contribution from the outside of the reservoir ($U \setminus R$) follows the same expression as the contribution from the reservoir (R). The total contribution to the gravimetric observations from the whole subsurface volume is thus the sum of the contributions of all single points:

$$\gamma \int_R \frac{\rho \frac{v_p}{2}}{r^2}\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}t + \gamma \int_{U \setminus R} \frac{\rho}{r^2}\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z \tag{2}$$

The integral over the volume outside of the reservoir is considered a confounding factor and we assume it is either constant or that the non-constant part has been removed in preprocessing of the gravimetric data for all vintages in a time sequence.

To linearize the expression above, Eq. (1), we do not express the vertical component z_i, in the distance measure r, in terms of the unknown values v_p. The vertical component z_i is determined as the integral of expected values of the prior distribution of v_p

$$z_i = \frac{1}{2} \int_0^{t_i} E(v_p)\,\mathrm{d}t \tag{3}$$

In the discrete case we let index $i = 0, \ldots, N_R$ run through all reservoir grid cells and the index $j = 0, \ldots, N_d$ run through all gravimetric observations points. We write the contribution of one point (x_i, y_i, z_i) to the gravimeter in position (x_j, y_j, z_j) as

$$g_{i,j} = \gamma \frac{\rho_i v_{p,i}}{2r^2}\,\mathrm{d}t\,\mathrm{d}x\,\mathrm{d}y = \frac{\gamma}{2} \frac{\rho_i v_{p,i}}{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \mathrm{d}t\,\mathrm{d}x\,\mathrm{d}y \tag{4}$$

We let d_j be the observation in the gravimeter located at position(x_j, y_j, z_j). The total contribution to this observation is the sum of the contributions of all reservoir grid cells

$$d_j = \sum_{i=0}^{N_R} g_{i,j}\rho_i v_{p,i} + g_{j,U\setminus R} + \epsilon_j \tag{5}$$

where ϵ_j is the observation error, $g_{j,U\setminus R}$ is the constant contribution from the outside environment of the reservoir, and the elements in a matrix $\mathbf{G}$ are given as

$$\mathbf{G}_{i,j} = \frac{\frac{\gamma}{2}(dx\, dy\, dt)}{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \tag{6}$$

4 Forward model in Gaussian-linear settings

We now consider all observations points and contributions from all grid cells in a linear expression. We set $\mathbf{d} = [d_1, \ldots, d_{N_D}]$ and $\boldsymbol{\epsilon} = [\epsilon_1, \ldots, \epsilon_{N_d}]$ and we define the seismic parameter $\mathbf{m} = [m_1, \ldots, m_{N_R}]$, $m_i = \rho_i v_{p,i}$. In Gaussian-linear settings the relation between data $\mathbf{d}$ and seismic parameters $\mathbf{m} = \boldsymbol{\rho}\mathbf{v_p}$ is described by

$$\mathbf{d} = \mathbf{G}\mathbf{m} + \mathbf{g}_c + \boldsymbol{\epsilon} \tag{7}$$

where the seismic parameter has a normal distribution

$$\mathbf{m} \sim \mathcal{N}(\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m) \tag{8}$$

and the error term has a normal distribution

$$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_\epsilon) \tag{9}$$

The time-independent constant contribution from the outside environment of the reservoir is given in $\mathbf{g_c}$ and $\mathbf{G}$ is a matrix with entries as described above. From the above linear expression, it follows that the data also has a normal distribution, that is $\mathbf{d} \sim N(\boldsymbol{\mu}_d, \boldsymbol{\Sigma}_d)$ with

$$\boldsymbol{\mu}_d = \mathbf{G}\boldsymbol{\mu}_m \tag{10}$$

$$\boldsymbol{\Sigma}_d = \boldsymbol{G}\boldsymbol{\Sigma}_m\boldsymbol{G}^T + \boldsymbol{\Sigma}_\epsilon \tag{11}$$

In the following calculations we eliminate the constant contribution $\boldsymbol{g}_c$ by considering the first observation $\mathbf{d^0}$ at time $t = 0$ as a base case with the following observations $\mathbf{d^k}$ as deviations from this base case, with $k = 1, \ldots, T$. We thus use $\Delta\mathbf{m} = \Delta(\boldsymbol{\rho}\boldsymbol{v}_p)$ as the unknown parameter. In the following superscripts denote time indices.

$$\mathbf{d}^0 = \mathbf{G}\mathbf{m}^0 + \mathbf{g}_c + \boldsymbol{\epsilon}^0 \tag{12}$$

$$\mathbf{d}^1 = \mathbf{G}\mathbf{m}^1 + \mathbf{g}_c + \boldsymbol{\epsilon}^1 \tag{13}$$

$$\mathbf{d}^1 - \mathbf{d}^0 = \mathbf{G}(\mathbf{m}^1 - \mathbf{m}^0) + (\boldsymbol{\epsilon}^1 - \boldsymbol{\epsilon}^0) = \mathbf{G}\Delta\mathbf{m}^1 + \Delta\boldsymbol{\epsilon}^1 \tag{14}$$

$$\mathbf{d}^{k+1} - \mathbf{d}^k = \mathbf{G}\Delta\mathbf{m}^{k+1} + \Delta\boldsymbol{\epsilon}^{k+1} \tag{15}$$

For ease of notation we only use $\mathbf{m}$ when referring to the unknown seismic parameter in the following, even though we consider the change in the parameter.

5 Conditional inverse problem

We assume that data observations at one time step depend only on the seismic parameters at the same time step. This is conditional independece and is expressed by the relation:

$$p(\mathbf{d}^k|\mathbf{m}^0,\ldots,\mathbf{m}^T) = p(\mathbf{d}^k|\mathbf{m}^k), \quad k = 1,\ldots,T \tag{16}$$

We want to compute the distributions of our unknown $\mathbf{m}^k$ conditioned by the gravimetric data $\mathbf{d}^k$:

$$p(\mathbf{m}^k|\mathbf{d}^k) \sim N(\boldsymbol{\mu}_{m^k|d^k}, \boldsymbol{\Sigma}_{m^k|d^k}) \tag{17}$$

By standard Bayes rule we have the two relations

$$\begin{aligned} p(\mathbf{d}^k|\mathbf{m}^k) &= \tfrac{p(\mathbf{d}^k,\,\mathbf{m}^k)}{p(\mathbf{m}^k)} \\ p(\mathbf{m}^k|\mathbf{d}^k) &= \tfrac{p(\mathbf{d}^k,\,\mathbf{m}^k)}{p(\mathbf{d}^k)} \end{aligned}$$

Combining these two relations gives us the conditional expression

$$p(\mathbf{m}^k|\mathbf{d}^k) = \frac{p(\mathbf{d}^k|\mathbf{m}^k)p(\mathbf{m}^k)}{p(\mathbf{d}^k)} \tag{18}$$

In Bayesian terms this can be written as a posterior model equal to a constant times the prior model multiplied with a likelihood model:

$$p(\mathbf{m}^k|\mathbf{d}^k) = \text{const} \cdot p(\mathbf{m}^k)p(\mathbf{d}^k|\mathbf{m}^k) \tag{19}$$

For the rest of the description we disregard the time superscript. Each inversion of gravity data is done independently at each vintage. We repeat the distributions for the seismic parameter

$$\mathbf{m} \sim \mathcal{N}(\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m) \tag{20}$$

and the error term

$$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_\epsilon) \tag{21}$$

From the linear relationship between data and the seismic parameters, Eq. (7), it follows that the data has the following distribution

$$\mathbf{d} \sim \mathcal{N}(\mathbf{G}\boldsymbol{\mu}_m,\ \mathbf{G}\boldsymbol{\Sigma}\mathbf{G}^T + \boldsymbol{\Sigma}_\epsilon) \tag{22}$$

The joint distribution of the seismic parameters and the data, is by standard terms given by

$$\begin{bmatrix} \mathbf{m} \\ \mathbf{d} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_m \\ \mathbf{G}\boldsymbol{\mu}_m \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_m & \boldsymbol{\Sigma}_m\mathbf{G}^T \\ \mathbf{G}\boldsymbol{\Sigma}_m^T & \mathbf{G}\boldsymbol{\Sigma}_m\mathbf{G}^T + \boldsymbol{\Sigma}_\epsilon \end{bmatrix} \right) \tag{23}$$

For better readability in the following expressions, we rewrite this as

$$\begin{bmatrix} \mathbf{m} \\ \mathbf{d} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_m \\ \mathbf{G}\boldsymbol{\mu}_m \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_m & \boldsymbol{\Sigma}_{md} \\ \boldsymbol{\Sigma}_{md}^T & \boldsymbol{\Sigma}_d \end{bmatrix} \right) \tag{24}$$

By standard results the conditional distribution for the seismic parameters given data is then

$$\mathbf{m}|\mathbf{d} \sim \mathcal{N}(\boldsymbol{\mu}_m + \boldsymbol{\Sigma}_{md}\boldsymbol{\Sigma}_d^{-1}(\mathbf{d} - \boldsymbol{\mu}_d),\ \boldsymbol{\Sigma}_m - \boldsymbol{\Sigma}_{md}\boldsymbol{\Sigma}_d^{-1}\boldsymbol{\Sigma}_m^T) \tag{25}$$

Thus the conditional mean and covariance is given by

$$\boldsymbol{\mu}_{m|d} = \boldsymbol{\mu}_m + \boldsymbol{\Sigma}_m \mathbf{G}^{\mathbf{T}} (\mathbf{G}\boldsymbol{\Sigma}_m \mathbf{G}^{\mathbf{T}} + \boldsymbol{\Sigma}_\epsilon)^{-1} (\mathbf{d} - \mathbf{G}\boldsymbol{\mu}_m) \quad (26)$$

$$\boldsymbol{\Sigma}_{m|d} = \boldsymbol{\Sigma}_m - \boldsymbol{\Sigma}_m \mathbf{G}^{\mathbf{T}} (\mathbf{G}\boldsymbol{\Sigma}_m \mathbf{G}^{\mathbf{T}} + \boldsymbol{\Sigma}_\epsilon)^{-1} \mathbf{G}\boldsymbol{\Sigma}_m \quad (27)$$

A central part in the computations of the mean and covariance is the expression $\mathbf{G}\boldsymbol{\Sigma}\mathbf{G}^{\mathbf{T}}$. This matrix product is of dimensions $N_d \times N_d$. However, since the matrix $\boldsymbol{\Sigma}$ has dimensions $N_R \times N_R$, the product involves nested sums over the number of grid cells N_R. With a potential large number of grid cells, this matrix multiplication is potentially computationally very expensive.

6 Upscaling of the conditonal inverse problem

To handle the computationally expensive matrix product we upscale the inverse problem to a coarse reservoir grid with fewer grid blocks. In the following we will use the term block for the collection of fine grid cells that make up a grid cell, that is a grid block, in the coarse grid. For ease of notation and explanation we upscale a regular grid. We denote the number of fine grid cells as N_R and the number of coarse grid blocks N_C. Each grid blocks consists of N_B fine grid cells, that is $N_C N_B = N_R$. The upscaling presented here is not restricted to regularly upscaled grids, although we do assume that the thickness of grid cells (the vertical distance) is uniform, or at least close to uniform.

In the current setting, we need to upscale the conditional inverse problem given by Eq. (26) and (27). The gravimetric observations given in $\mathbf{d}$ are kept unchanged while the quantities related to the reservoir model are upscaled. More particularly, we need to upscale the prior mean $\boldsymbol{\mu}_m$, prior covariance matrix $\boldsymbol{\Sigma}_m$ and the gravity matrix $\mathbf{G}$. Upscaling the latter matrix follows straightforward upscaling based on coarse grid quantities. The stochastic variable $\mathbf{m}$ is upscaled by smoothing and subsampling. The following sections will give both explanations through formulas as well as examples.

6.1 Upscaling the gravity matrix

The gravity matrix $\mathbf{G}$ is upscaled directly by calculating the entries in Eq. 6 with coarse grid block dimensions and coarse grid block center positions. The observation positions running over index j are the same. We thus get the expression

$$\mathbf{G}_{I,j} = \frac{\frac{\gamma}{2}(\mathrm{d}X\,\mathrm{d}Y\,\mathrm{d}T)}{(x_I - x_j)^2 + (y_I - y_j)^2 + (z_I - z_j)^2} \quad (28)$$

with coarse indices I to denote grid block indices in the upscaled grid and capitalization of $\mathrm{d}X$, $\mathrm{d}Y$ and $\mathrm{d}T$ to denote coarse grid quantities.

To consider the error introduced in the coarse expression compared to the fine grid expression, we assume for now that the density is constant within a coarse grid cell. This a good basis for comparision since the density change has high continuity. We consider the gravimetric contribution of one coarse grid block to one observation point using the fine scale grid:

$$\sum_{i=1}^{N_B} \gamma \frac{m_i}{r_{i,j}^2} = \sum_{i=1}^{N_B} \gamma \frac{(\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z)\rho}{r_{i,j}^2} = \gamma(\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z)\rho \sum_{i=1}^{N_B} \frac{1}{r_{i,j}^2} \quad (29)$$

Here the subscript i denotes fine grid quantities. For a regular grid we have constant grid cell dimensions, that is $\mathrm{d}x_i = \mathrm{d}x$, $\mathrm{d}y_i = \mathrm{d}y$, $\mathrm{d}z_i = \mathrm{d}z$. We write out the corresponding expression for

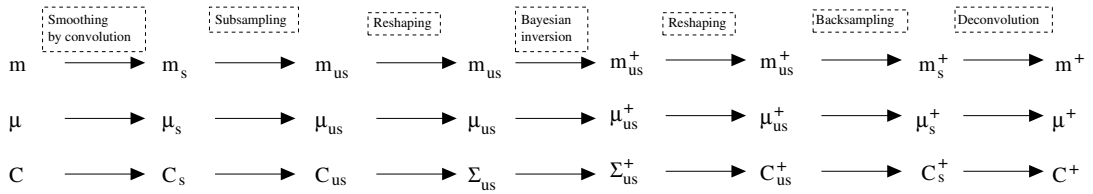


Figure 2. Mnemonic illustration of upscaling of a stochastic parameter, introducing subscripts and terms for the operations.

the coarse grid block with index I, using the coarse grid quantities:

$$\gamma \frac{m_I \rho}{r_{I,j}^2} = \gamma \frac{S \mathrm{d}x\, \mathrm{d}y\, \mathrm{d}z \rho}{r_{I,j}^2} = \gamma S (\mathrm{d}x\, \mathrm{d}y\, \mathrm{d}z) \rho \frac{1}{r_{I,j}^2} \tag{30}$$

Here the coarse grid block dimensions give

$$\mathrm{d}X\, \mathrm{d}Y\, \mathrm{d}Z = \mathrm{d}x N_{BX}\, \mathrm{d}y N_{BY}\, \mathrm{d}z N_{BZ} = N_B \mathrm{d}x\, \mathrm{d}y\, \mathrm{d}z$$

with the scaling factor $N_{BX} N_{BY} N_{BZ} = N_B$ and r_I is the distance from the center point in the coarse grid block I to the observation point. In the upscaling we thus approximate the above fine scale sum of the inverse square distances with the coarse scale inverse square distance:

$$\sum_{i=1}^{N_B} \frac{1}{r_{i,j}^2} \approx N_B \frac{1}{r_{I,j}^2} \tag{31}$$

For the cases where the coarse distance is a good approximation to the sum of fine scale distances, we introduce a small upscaling error. For cases where the coarse distance is not a good approximation, we introduce potentially significant upscaling error.

One possible refinement in the upscaled gravity matrix $\boldsymbol{G}$ is to keep the fine grid resolution of the distance measure instead of using the upscaled distances.

$$r_{I,j}^2 = \frac{1}{N_B} \frac{1}{\sum_{i=1}^{N_B} \frac{1}{r_{i,j}^2}} \tag{32}$$

The square of the coarse distance corresponds to the harmonic average of the squares of fine-scale distances.

Exemplification of the upscaling of coarse grid quantities will be included in the MATLAB example in Section 7. We do not consider exactly upscaling of the gravity matrix. We rather consider the gravimetric response that a coarse grid gives. This involves computing the same quantities as in the gravity matrix, summed up for each row to give the contribution to each gravimetric observation point.

6.2 Upscaling the stochastic seismic parameter

Upscaling the stochastic seismic parameter $\mathbf{m}$ means upscaling the mean and covariance function describing the normal distribution

$$\mathbf{m} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{C}) \tag{33}$$

To describe the upscaling we start with a mnemonic illustration in Figure 2 to give a overview of the required steps and the subscript notation that will be used. As Figure 2 schematizes, we separate the upscaling into two steps:

1. Smoothing by use of convolution

2. Sub-sampling to reduce size of cube/parameter

The reshaping step in the illustration refers to the change in representation of the parameter. We vary between three dimensional cubes, and vectors and two-dimensional matrices. This is not included as a step in the upscaling process, as it is a matter of representing the variable, regardless of being upscaled or not.

The inverse operation to this upscaling has the corresponding two steps:

1. Backsampling the sub-cube into the smoothed fine-scale cube
2. Inverse smoothing by use of pseudo-deconvolution

We introduce the term backsampling instead of the conventional term upsampling from signal processing, since we do this backsampling in the Fourier domain and use a previous variable in the upscaling chain as basis into which we put back upscaled and conditioned frequencies.

We start with describing upscaling of the mean. For the sake of completness, subsampling both in the Fourier domain and in the real domain is described. The reverse process of smoothing and backsampling is subsequently described. Vectorization of the mean is carried out as a straightforward reshaping from a three dimensional cube to a vector. This will not be described in any more detail. Upscaling steps for the covariance function are generally identical to the ones for the mean, and therefore only repeated in short. Steps that are different than for the mean, as in the case of double convolving the covariance function and reshaping with respect to circularity, is described in more detail.

All calculation using the Fourier transform can be performed using the Fast Fourier Transform (FFT).

Smoothing the prior mean

For ease of notation we neglect subscript m from Eq. (26) and (27) and introduce subscripts describing in which state in the upscaling process the variable currently is: s for smoothed and us for upscaled (realized as subsampled from the smoothed variable). Figure 2 uses this notation.

We want to average cells in a coarse grid block I

$$\begin{aligned} \overline{\mu_I} &= \frac{1}{N_B} \sum_{i \in B(I)} \mu_i, \quad I = 1, \ldots, N_C \\ B(I) &= \{i : \text{Fine cell } i \text{ is contained in coarse block } I\}. \end{aligned} \tag{34}$$

In words this is the averaged sum over the fine grid cells that make up coarse block I. For the sake of completness, we write the above sum with $\boldsymbol{\mu}$ being represented as a three dimensional cube of dimensions $N_x \times N_y \times N_z$. Hence, the block $B(I)$ is a cube of dimensions $N_{BX} \times N_{BY} \times N_{BZ}$, where N_{BX} denotes the number of fine grid cells in the x direction the block consists of, and N_{BY} and N_{BZ} denotes the number of fine grid cells in direction y and z, respectively. The coarse grid is regularly subdivided into $N_{CX} \times N_{CY} \times N_{CZ}$ blocks. We thus have the relations

$$N_{CX} N_{BX} = N_x, \quad N_{CY} N_{BY} = N_y, \quad N_{CZ} N_{BZ} = N_z.$$

The averaging sum can be written as

$$\overline{\mu_{l_1,l_2,l_3}} = \frac{1}{N_B} \sum_{i=1}^{N_{BX}} \sum_{j=1}^{N_{BY}} \sum_{k=1}^{N_{BZ}} \mu_{(l_1+i-1),(l_2+j-1),(l_3+k-1)} \tag{35}$$

with the three indices indicating the cubic representation. Local averaging for all coarse grid blocks can be written as a sum corresponding to convolution with an appropriate convolution

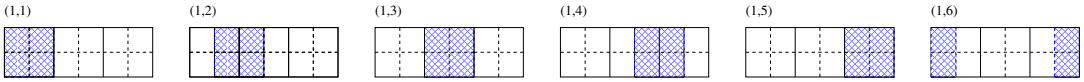


Figure 3. Convolution process on a 2×6 grid (dotted lines). Coarse grid of dimension 1×3 is outlined with full lines. The subset of configuration that corresponds to coarse grid blocks consists of the configurations $(1,1)$, $(1,3)$ and $(1,5)$, hence the first fine grid cell in the grouping of cells to a coarse grid block.

kernel:

$$\begin{gathered}\overline{\mu_{l_1,l_2,l_3}} = \sum_{i=1}^{N_{BX}} \sum_{j=1}^{N_{BY}} \sum_{k=1}^{N_{BZ}} \mu_{(l_1+i-1),(l_2+j-1),(l_3+k-1)} h_{i,j,k}, \\ \text{for } l_1 = 1,\ldots,N_{CX}, l_2 = 1,\ldots,N_{CY}, l_3 = 1,\ldots,N_{CZ}\end{gathered} \tag{36}$$

The convolution kernel that we use is defined as

$$h_{i,j,k} = \begin{cases} \frac{1}{N_B} & \text{if } (i,j,k) \text{ is a cell in the first coarse grid block } B(1) \\ 0 & \text{otherwise} \end{cases} \tag{37}$$

This function can be viewed as a block filter or moving average, and when used in convolution corresponds to taking the local average of the fine grid cells within the coarse blocks that is defined by the function.

For ease of notation, we introduce the multi-index $\boldsymbol{i} = (i,j,k)$ and $\boldsymbol{l} = (l_1,l_2,l_3)$ and write

$$\overline{\mu_{\boldsymbol{l}}} = \sum_{\boldsymbol{i}=1}^{\boldsymbol{N}} \mu_{\boldsymbol{l}-\boldsymbol{i}} h_{\boldsymbol{i}} \equiv \mu * h \tag{38}$$

We thus define smoothing the prior mean as the convolution of the mean with the function $\boldsymbol{h}$:

$$\boldsymbol{\mu}_s = \boldsymbol{\mu} * \boldsymbol{h} \tag{39}$$

Transforming the convolution to the Fourier domain, let us take advantage of the convolution theorem stating that the Fourier transform of a convolution is the point wise product of the Fourier transforms:

$$\mathcal{F}\{f * g\} = \mathcal{F}\{f\} \cdot \mathcal{F}\{g\} \tag{40}$$

Thus the smoothing step can be expressed as

$$\begin{aligned}\mathcal{F}\{\boldsymbol{\mu}_s\} &= \mathcal{F}\{\boldsymbol{\mu}\} \cdot \overline{\mathcal{F}\{\boldsymbol{h}\}} \\ \boldsymbol{\mu}_s &= \mathcal{F}^{-1}\{\mathcal{F}\{\boldsymbol{\mu}\} \cdot \overline{\mathcal{F}\{\boldsymbol{h}\}}\}\end{aligned} \tag{41}$$

with complex conjugate denoted with overline over the variable.

Subsampling the prior mean

To reduce the size of a smoothed variable we subsample it. In signal processing the conventional term is downsample. To indicate that the process we perform is to reduce a full cube into a subcube, we rather apply the term subsampling.

In the real domain subsampling a smoothed variable may be visualized by Figure 3. Here we can imagine the blue region (corresonding to the convolving kernel h) to slide across the 2×6 grid (outlined with dashed lines) forming a 2×6 smoothed grid. This grid is furthermore coarsened to a 1×3 grid (full lines) and the blue region thus corresponds to the first coarse grid block. As the blue block slides across the grid, we observe that only every second configuration corresponds to a averaging of a coarse grid block. The other configurations correspond to averaging fine grid

cells from two coarse blocks and are not of interest in this case. Thus subsampling to obtain a coarse grid coresonds to selecting every element in the full cube corresponding to the first fine-grid cell in a coarse grid block. In other words, in the real domain subsampling corresponds to picking elements from the variable with a certain interval. Using a MATLAB-style notation for subset of indices, this will be for our three-dimensional cube the following indices:

$$[1 : N_{BX} : N_x,\ 1 : N_{BY} : N_y,\ 1 : N_{BZ} : N_z] \tag{42}$$

On the other hand, subsampling in the Fourier domain corresponds to filtering out high-frequent information. This is because smoothing in the Fourier domain can be considered as a keeping information in the low frequencies, while high frequencies are disregarded. One subsampling scheme is therefore to only consider the low frequencies in the Fourier domain.

Explicit exemplification of convolution and subsampling in one dimension is included in Appendix B.

Note that in practical use the convolving kernel given in Eq. (37) together with the subsampling introduce scaling into the expressions. This is because the forward and inverse Fourier transforms work on different grids. In the expression below the forward Fourier transform applied on $\boldsymbol{\mu}_s$ works on a grid of original dimensions, while the inverse Fourier transform in this case works on the upscaled grid. In MATLAB to obtain correct scale of the inverse Fourier transform of the smoothed and subsampled mean, we scale with the inverse of the upscaling factor:

$$\boldsymbol{\mu}_{us} = \frac{1}{N_B}\mathcal{F}^{-1}\{\mathcal{F}\{\boldsymbol{\mu}_s\}(\omega_{low})\} \tag{43}$$

This scaling factor is dependent on the formulation of the transform used.

Backsampling the posterior mean

We now assume that Bayesian inversion or some other changes have been done to the variable and refer now to the variable as the *posterior* mean. Figure 2 introduces superscript $+$ used for denoting a change in the varibable (conditioning from Bayesian inversion). Hence $\boldsymbol{\mu}_{us}^{+}$ denotes upscaled posterior mean.

In the current context, we denote the reverse process of subsampling as backsampling. In signal processing the conventional term is upsampling. The term backsampling is meant to denote the process of using the smoothed prior variable and replace or put back possibly updated low frequencies in the Fourier domain. These low frequencies are the ones that have been kept for the upscaled variable. Backsampling is thus meant to refer to the process in the Fourier domain of updating some frequencies in the smoothed prior mean and then putting them back into the original variable which now becomes the posterior. The ulitmate purpose is to extend the subsampled (upscaled) variable to the original size.

$$\begin{aligned}\mathcal{F}\{\boldsymbol{\mu}_s^{+}\}(\omega_{high}) &= \mathcal{F}\{\boldsymbol{\mu}_s\}(\omega_{high})\\ \mathcal{F}\{\boldsymbol{\mu}_s^{+}\}(\omega_{low}) &= \mathcal{F}\{\boldsymbol{\mu}_{us}^{+}\}(\omega_{low})\end{aligned} \tag{44}$$

where ω_{low} are the low frequencies selected in the subsampling.

In the real domain this backsampling is better viewed as upsampling, as it corresponds to standard upsampling in signal processing. In other works, upsampling in the real domain means increasing the elements of a variable by inserting interpolated or kriged values in between the upscaled values, to extend - upsample - the variable.

As written above, due to the fact that the Fourier transforms work on different grids, we scale the upscaled prior mean after taking the inverse Fourier transform of the subsampled variabale in the

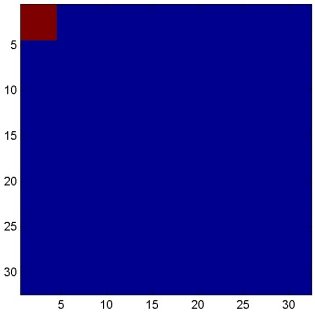


Figure 4. Slice of the convolution kernel. Blue equals 0.

Fourier domain, Eq. (43). Likewise, returning to the Fourier domain from the upscaled posterior mean, we need to remove this scaling. We assume now that the change done to the upscaled variable has been done in the real domain. In MATLAB the appropriate scale is to multipy with N_B:

$$\mathcal{F}\{\boldsymbol{\mu}_{us}^{+}\} = \mathcal{F}\{\boldsymbol{\mu}_{us}^{+} \cdot N_B\} \tag{45}$$

Reverse smoothing of the posterior mean

The reverse smoothing operation is performed as pseudo-deconvolution defined as pointwise division in the Fourier domain:

$$\mathcal{F}\{\boldsymbol{\mu}^{+}\} = \mathcal{F}\{\boldsymbol{\mu}_{s}^{+}\}./\mathcal{F}\{\boldsymbol{h}\} \tag{46}$$

As large parts of $\mathcal{F}\{\boldsymbol{h}\}$ consists of 0, special treatment for division by 0 has to be done. For the elements where $\mathcal{F}\{\boldsymbol{h}\} = 0$ we simply insert the value of the Fourier transform of the prior mean corresponding to the current frequency ω:

$$\mathcal{F}\{\boldsymbol{\mu}^{+}\}(\omega) = \mathcal{F}\{\boldsymbol{\mu}\}(\omega) \tag{47}$$

We have referred to the process as *pseudo*-deconvolution due to the special treatment for handling 0's in the convolution kernel. However, it is actually nothing else than Bayesian update, since zero in the kernel means no information added.

MATLAB example of upscaling mean

Firstly, a simple and intuitive one-dimensional example is included in Appendix B to document practical use of the steps described above. Here we include an example more directed toward use in gravimetric inversion, that is we illustrate the significant steps needed for upscaling the mean of a stochastic parameter. The two examples may unfortunately appear as repetetative and thus tedious. However, realization of the upscaling in MATLAB has been a good source of verifying the logic of the upscaling and are therefore included for the sake of complete documentation and later reference.

We assume a 3D random Gaussian field, called `mu`, with dimensions $32 \times 32 \times 32$, upscaled to a $8 \times 8 \times 8$ coarse grid. The leftmost plot in Figure 5 shows a slice of the variable.

We introduce an averaging convolving kernel, corresponding to values in the first coarse block, and zero elsewhere. Figure 4 shows a 2D slice of the function, with blue region equalling 0 and red a constant.

We start the upscaling by applying the Fourier transform on the random variable and the convolution kernel

```
fft_mu = fftn(mu);
fft_h  = fftn(h);
```

Applying the convolution theorem let us do convolution as point wise multiplication in the Fourier domain. Note that we use the complex conjugate of the convolution kernel. The second left plot in Figure 5 shows the result of this smoothing step.

```
fft_mu_s  = fft_mu.*conj(fft_h);
```

Subsampling in the Fourier domain corresponds to keeping the low frequencies. Here we consider the eight subcubes in the corners, collapsed into one subsampled variable. The third plot from the left side in Figure 5 shows a slice of the subsampled variable.

```
%fft_mu_us = fft_mu_s([1:nx_up/2, (nx-(nx_up/2)+1):nx], ...
%                     [1:ny_up/2, (ny-(ny_up/2)+1):ny], ...
%                     [1:nz_up/2, (nz-(nz_up/2)+1):nz]);
i_start  = 1:nx_up/2;
j_start  = 1:ny_up/2;
k_start  = 1:nz_up/2;
i_end    = nx-nx_up/2+1:nx;
j_end    = ny-ny_up/2+1:ny;
k_end    = nz-nz_up/2+1:nz;
i_end_us = nx_up-nx_up/2+1:nx_up;
j_end_us = ny_up-ny_up/2+1:ny_up;
k_end_us = nz_up-nz_up/2+1:nz_up;
fft_mu_us = zeros(nx_up, ny_up, nz_up);

fft_mu_us(i_start, j_start, k_start)  = fft_mu_s(i_start,j_start,k_start);
fft_mu_us(i_end_us,j_start, k_start)  = fft_mu_s(i_end,  j_start,k_start);
fft_mu_us(i_start, j_end_us,k_start)  = fft_mu_s(i_start,j_end,  k_start);
fft_mu_us(i_end_us,j_end_us,k_start)  = fft_mu_s(i_end,  j_end,  k_start);
fft_mu_us(i_start, j_start, k_end_us) = fft_mu_s(i_start,j_start,k_end);
fft_mu_us(i_end_us,j_start, k_end_us) = fft_mu_s(i_end,  j_start,k_end);
fft_mu_us(i_start, j_end_us,k_end_us) = fft_mu_s(i_start,j_end,  k_end);
fft_mu_us(i_end_us,j_end_us,k_end_us) = fft_mu_s(i_end,  j_end,  k_end);
```

When taking the inverse Fourier transform of the upscaled variable, we only keep the real component. Note the scaling of the transformed variable.

```
mu_us = real(ifftn(fft_mu_us));
mu_us = mu_us*(nx_up*ny_up*nz_up)/(nx*ny*nz);
```

The alternative subsampling is in the real domain. Note no scaling here.

```
mu_s   = real(ifftn(fft_mu_s));
mu_us2 = mu_s(1:(nx/nx_up):nx, 1:(ny/ny_up):ny, 1:(nz/nz_up):nz);

fft_mu_us2 = real(fftn(mu_us2));
```

For the purpose of a simple example, we change the upscaled variable by adding the constant 10 to the whole field. We point out that this has no relation to Bayesian inversion, which is the change we are interested in in the actual gravimetric inversion. The third plot from the right side in Figure 5 shows a slice of the changed upscaled variable. Note the change of scale in the plot from the previous step.

```
mu_us_changed = mu_us + 10;
```

In the reverse process of upscaling, we Fourier transform the upscaled and changed random variable. Note that we remove the scaling introduced of the inverse Fourier transformed upscaled variable.

```
fft_mu_us = fftn(mu_us_changed*(nx*ny*nz)/(nx_up*ny_up*nz_up));
```

We put back the 8 subcubes making up the upscaled variable into the 8 corners of the original smoothed variable. We reuse the indices from above. The second right plot in Figure 5 shows the backsampled variable.

```
fft_mu2_s = fft_mu_s; % Copy smoothed variable, original size
fft_mu2_s(i_start, j_start, k_start) = fft_mu_us(i_start,  j_start,  k_start);
fft_mu2_s(i_end,   j_start, k_start) = fft_mu_us(i_end_us, j_start,  k_start);
fft_mu2_s(i_start, j_end,   k_start) = fft_mu_us(i_start,  j_end_us, k_start);
fft_mu2_s(i_end,   j_end,   k_start) = fft_mu_us(i_end_us, j_end_us, k_start);
fft_mu2_s(i_start, j_start, k_end)   = fft_mu_us(i_start,  j_start,  k_end_us);
fft_mu2_s(i_end,   j_start, k_end)   = fft_mu_us(i_end_us, j_start,  k_end_us);
fft_mu2_s(i_start, j_end,   k_end)   = fft_mu_us(i_start,  j_end_us, k_end_us);
fft_mu2_s(i_end,   j_end,   k_end)   = fft_mu_us(i_end_us, j_end_us, k_end_us);
mu2_s = real(ifftn(fft_mu2_s));
```

Next, we carry out pseudo-deconvolution. Note the special treatment where the Fourier transform of the convolving kernel is 0. For those frequencies we keep the original values from Fourier transform of the original variable. Note also the use of the complex conjugate of the Fourier tranform of the convolution kernel.

```
fft_mu2 =  fft_mu2_s./conj(fft_h);
I = find(fft_h==0);   % Handle zeros in the fft_h properly
fft_mu2(I) = fft_mu(I);
```

Lastly, we take the inverse Fourier tranform of the backsampled and deconvolved variable to find the fine-scale representation of the changed variable.

```
mu2 = real(ifftn(fft_mu2));
```

The rightmost plot in Figure 5 shows the final result of the upscaling.

The desired difference between the final variable and the original variable is the constant added to the upscaled random variable.

```
mean(mean(mean(mu2-mu)))

ans =
   10.0000
```

Note that this difference is only a check of correctness of the process and is a direct result of the change in the example, that was addition of constant 10 to the whole field.

Figure 5 shows 2D slices of the variables at the different steps. Note the change in scale between the two middle plots. This change in scale reflects the addition of the constant 10.

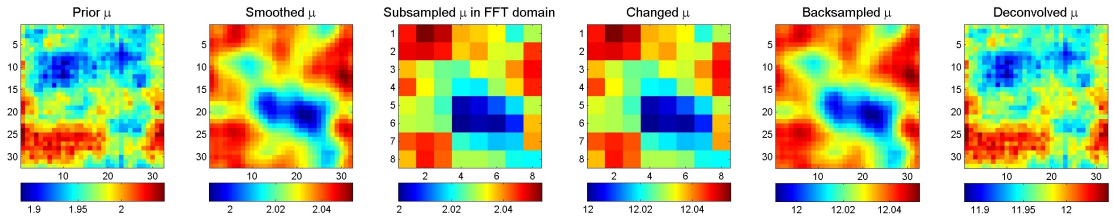


Figure 5. Upscaling μ in MATLAB. The plots show 2D slices at the different steps in the upscaling. Note the change in scale between the two middle plots, that is the two upscaled variables.

Smoothing of the prior covariance function

For upscaling the covariance matrix $\mathbf{\Sigma}_m$ in Eq. (26) and (27), we use double convolution of the covariance function with the same convolution kernel as for the mean, reflecting local averaging. We assume the covariance to be stationary meaning that the standard deviation of the seismic parameters $\mathbf{m}$ is constant, that is $\sigma(\mathbf{x}) = \sigma$ for all $\mathbf{x}$. Considering the fine-scale reservoir grid, we can express the covariance between $\mathbf{m}(\boldsymbol{x}_i)$ and $\boldsymbol{m}(\boldsymbol{x}_j)$ as a function c of the distance between the points $\mathbf{x}_i$ and $\mathbf{x}_j$:

$$\mathbf{C}(\mathbf{m}(\mathbf{x}_i), \mathbf{m}(\mathbf{x}_j)) = \mathbf{C}(\mathbf{x}_i, \mathbf{x}_j) = \sigma(\mathbf{x}_i)\sigma(\mathbf{x}_j)\rho(\mathbf{x}_i - \mathbf{x}_j) = c(\mathbf{x}_i - \mathbf{x}_j) = c(\mathbf{x}_i, \mathbf{x}_j) \tag{48}$$

Here $\sigma(\mathbf{x}_i)$ is the standard deviation of $\mathbf{m}$ and $\rho(\mathbf{x}_i - \mathbf{x}_j)$ is the correlation coefficient between $\mathbf{m}(\mathbf{x}_i)$ and $\mathbf{m}(\mathbf{x}_j)$ only depending on the distance between the two points. The distance $(x_i - x_j)$ is what is later referred to as *lag*.

We define smoothing of the covariance given by the function $c(x_i - x_j)$ as a convolution:

$$\bar{c}(x_i, x_j) = \int c(x_i, x)h(x_j - x)\, dx = \sigma^2[h * \rho](x_i, x_j) \tag{49}$$

where we define the function $h(\mathbf{x})$ as an averaging function, identically as in Eq. (37):

$$h(x) = \begin{cases} \frac{1}{N_B} & \text{if } x \text{ is within the first coarse grid block} \\ 0 & \text{otherwise.} \end{cases} \tag{50}$$

For the double convolved covariance we have

$$\bar{\bar{c}}(x_i, x_j) = \int \bar{c}(x, x_j) h(x - x_i) dx = \sigma^2 [h * \rho * h](x_i, x_j) \tag{51}$$

Applying the convolution theorem on the double convolved covariance gives pointwise multiplication in the Fourier domain:

$$\bar{\bar{c}}(x_i, x_j) = \sigma^2 \mathcal{F}^{-1} \cdot [\mathcal{F}\{h\} \cdot \mathcal{F}\{\rho\} \cdot \overline{\mathcal{F}\{h\}}](x_i, x_j) \tag{52}$$

Note the overline above the second Fourier tranform of the kernel which denotes the complex conjugate.

This is then the smoothed covariance function we are interested in:

$$\boldsymbol{C}_s = \bar{\bar{c}}(x_i, x_j) \tag{53}$$

We introduce spectral density $\tilde{\rho}$ as the Fourier transform of the correlation function ρ:

$$\mathcal{F}\{\rho\}(\omega) = \tilde{\rho}(\omega) \tag{54}$$

Subsampling the prior covariance function

Subsampling the covariance function follows the same procedure as described for the mean. In the real domain, subsampling consists of selecting values from the smoothed variable corresponding to the first cell in each coarse grid block. Subsampling in the frequency domain corresponds to keeping the low frequencies only. Formulas are identical as for the mean, thus not repeated here.

Backsampling with truncation of the posterior covariance function

Likewise as for the subsampling, the backsampling process is the same for the covariance function as for the mean. Again, the formulas are identical as for the mean, thus not restated here.

Reverse smoothing of the posterior covariance function

The reverse smoothing operation on the covariance function is performed as double pseudo-deconvolution defined as pointwise division in the Fourier domain:

$$\mathcal{F}\{\boldsymbol{C}^+\} = \mathcal{F}\{\boldsymbol{C}_s^+\}./\left(\mathcal{F}\{\boldsymbol{h}\}\overline{\mathcal{F}\{\boldsymbol{h}\}}\right) \tag{55}$$

Note again the complex conjugate of the Fourier transform of the kernel.

Again as for the deconvolution of the mean, we include special treatment for division by 0. For the elements where $\mathcal{F}\{\boldsymbol{h}\} = 0$ we again insert the value of the Fourier transform of the prior covariance corresponding to the current frequency ω:

$$\mathcal{F}\{\boldsymbol{C}^+\}(\omega) = \mathcal{F}\{\boldsymbol{C}\}(\omega) \tag{56}$$

Reshaping of the covariance function to a covariance matrix

Reshaping of the mean is a straightforward vectorization of a three dimensional cube. Reshaping the three-dimensional upscaled covariance matrix to a two-dimensional matrix on the other hand, needs more consideration.

First of all, note the dimensions of the covariance function and covariance matrix: The upscaled covariance function forms a 3D matrix (often referred to as cube) of size $N_{CX} \times N_{CY} \times N_{CZ}$ and reflects the covariance of one position to all the other positions in the grid. The matrix product in the conditional inversion expressions (Eq. (26) and (27)) requires a covariance matrix of size $N_C \times N_C$. The covariance matrix reflects the covariance between two positions in the reservoir grid. In other words, the element in the covariance matrix in position (i, j) is the covariance between position i and j.

The reshaping of the function to the matrix is done with respect to the lag, that is the distance of one point to another point. Computation of the lag index is sketched in Algorithm 1 and actual MATLAB code is attached in Appendix C.1.

According to the lag index, values from the covariance function is spread out to one or more positions in the covariance matrix. This process is sketched out in Algorithm 2 and actual MATLAB code is attached in Appendix C.2.

The reverse process of forming a covariance function from the covariance matrix, consists of looping through the set of all lag indices and for each set average all contributions in the covariance matrix that correspond to this lag index. Algorithm 3 sketches this process and actual MATLAB code is in Appendix C.3.

MATLAB example of upscaling covariance

We generate a covariance function of dimensions $64 \times 64 \times 64$ cells and upscale to a $8 \times 8 \times 8$ coarse grid. The top leftmost plot in Figure 7 shows a 2D slice of the covariance function.

We define the lag as the index distance between two cells. Here we include an example of lag index grid for a $4 \times 4 \times 4$ grid in Figure 6. Colors indicate the numbers from 1 to 4. White color/no color means not defined lag index.

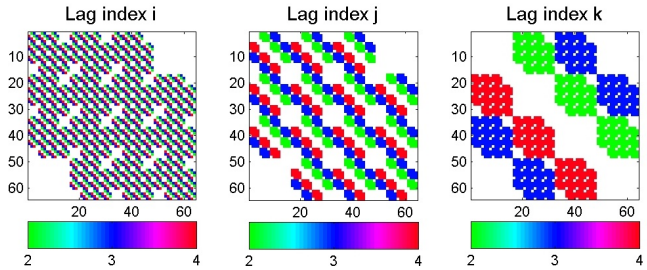


Figure 6. Example of the indices defining the lag index grid for a $4 \times 4 \times 4$ grid. The colors indicate the numbers from 1 to 4. No color means not defined lag index.

Smoothing the covariance function is done by double convolution. Here we include the MATLAB code for this step. Note the use of absolute value for the Fourier tranform of the convolution kernel.

```
fft_covFunc3D_s = fft_covFunc3D.*abs(fft_h).^2;
```

Second left plot in the top row in Figure 7 shows a slice of this smoothed variable.

We subsample and scale the subsampled variable as in the case for upscaling of the mean. The second right plot in the top row in Figure 7 shows the slice of this variable. Note no change in scale from the smoothed variable, and note the reduction in number of cells.

Reshaping of the covariance function to a two-dimensional covariance matrix is done according to the algorithmic descriptions in the above section. The matrix is shown in the top rightmost plot

for cell (i_1, j_1, k_1) **do**
  Define global index I
  for cell (i_2, j_2, k_2) **do**
    Define global index J
    Define lag: $(\Delta i, \Delta j, \Delta k) = (i_2, j_2, k_2) - (i_1, j_1, k_1)$
    if $|(\Delta i, \Delta j, \Delta k)| \leq (\frac{n_x}{2}, \frac{n_y}{2}, \frac{n_z}{2})$ **then**
      if $(\Delta i, \Delta j, \Delta k) > (0, 0, 0)$ **then**
        Lag index for $(I, J) = (\Delta i + 1, \Delta j + 1, \Delta k + 1)$
      else
        Lag index for $(I, J) = (n_x + \Delta i + 1, n_y + \Delta j + 1, n_z + \Delta k + 1)$
      end if
    else
      Lag index for (I, J) undefined
    end if
  end for
end for

Algorithm 1. Find lag index

for cell (i_1, j_1, k_1) **do**
  Define global index I
  for cell (i_2, j_2, k_2) **do**
    Define global index J
    if Lag index of(I, J) is defined **then**
      Covariance matrix position (I, J) = element from covariance function in lag index for (I, J)
    else
      Covariance matrix position $(I, J) = 0$
    end if
  end for
end for

Algorithm 2. Reshape according to lag

for cell (i_1, j_1, k_1) **do**
  Define global index I
  for cell (i_2, j_2, k_2) **do**
    Define global index J
    if Lag index of (I, J) defined **then**
      Sum(lag index) += cov matrix(I, J)
      Count(lag index) += 1
    end if
  end for
end for
for cell (i_1, j_1, k_1) **do**
  covariance function (i_1, j_1, k_1) = sum(i_1, j_1, k_1)/count(i_1, j_1, k_1)
end for

Algorithm 3. Reshape back to 3D covariance function

in Figure 7. Note the elements are of the same scale as the upscaled covariance. For the interested reader we include the MATLAB code of the functions in Appendix C.2 and C.3

The reshaped covariance matrix is now denoted `Sigma` in the MATLAB code. For the purpose of a simple example of upscaling, we introduce a simple modification of the variance: A reduction of the variance to half of the origianl values. We point out that in an actual case of upscaling the covariance, the modification of the covariance will be Bayesian inversion.

```
Sigma_changed = Sigma*0.5;
```

This matrix is shown in the lowest leftmost plot in Figure 7. Note the change of scale from the top rightmost plot, reflecting that the changed covariance matrix has reduced the values to half of the original values.

Likewise as for reshaping from covariance function to covariance matrix, the reverse reshaping is described in algorithmic form above and we only include the MATLAB functions in Appendix C.3. The 2D slice is shown in the second left in the lower row in Figure 7.

The reverse process of upscaling includes again removing scaling of the upscaled (and modified) covariance function before applying the inverse Fourier transform, with subsequent backsampling in the Fourier domain. This is done as for the mean. The backsampled slice is shown in the second right plot in the lower row in Figure 7.

Lastly, the reverse smoothing operation is done as a double pseudo-deconvolution. Again, note the use of absolute value of the Fourier transformed convolution kernel.

```
fft_covFunc3D_ss =  fft_covFunc3D_backsampled./abs(fft_h.^2);
I = find(fft_h==0);   % Handle zeros in the fft_indexGrid properly
fft_covFunc3D_ss(I) = fft_covFunc3D(I);

covFunc3D_ss = real(ifftn(fft_covFunc3D_ss));
```

The rightmost lower plot in Figure 7 shows the final variable.

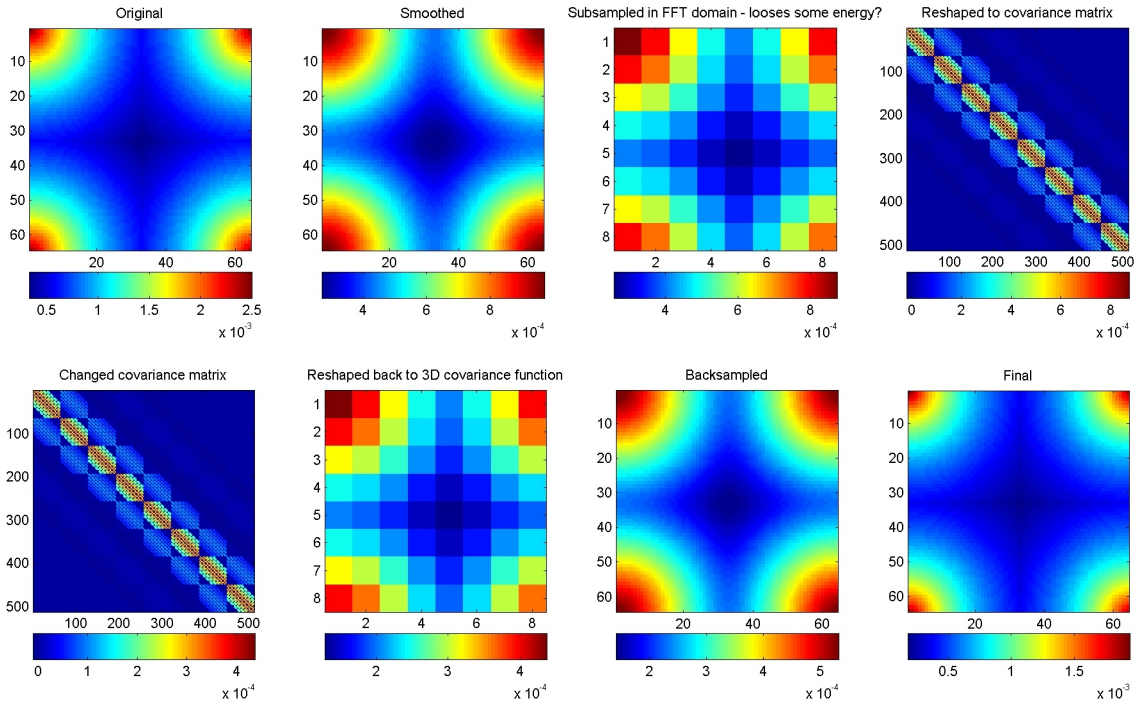


Figure 7. 2D slices of the covariance at the different steps in the upscaling process. Note different resolutions and scales of the colorbars of the plots.

6.3 Adjustment of the upscaled inverse problem to a positive definite system

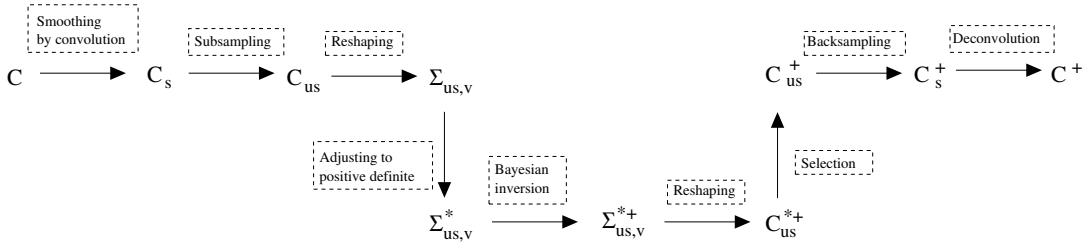


Figure 8. Mnemonic illustration of the process of adjusting the covariance matrix. Superscript star denotes adjustment to the matrix, while superscript + denotes conditioning by Bayesian inversion.

In Figure 2 the mnemonic sketch indicates Bayesian inversion of the upscaled covariance matrix with no other changes than just reshaping. However, the covariance function is defined on a cyclic domain so when the covariance matrix is created lags extending half of the padded grid are set to zero. This might result in a covariance matix which is not positive definite.

Hence, Figure 8 illustrates the detour in the upscaling chain to obtain positive definiteness and the way back to the downscaled steps needed for a complete solution. This subsection will describe the two added steps of adjusting the upscaled inverse problem to a positive definite system and reverting the effect to obtain "true" inverted covariance function. We point out that these adjustments described here are tailored for use with upscaled Bayesian inversion of gravimetric data, due to the low frequent content of the data.

For adjusting the upscaled covariance matrix $\mathbf{\Sigma}_{us,v}$ to a positive definite matrix, we first find the eigendecomposition

$$\mathbf{\Sigma}_{us,v} = \boldsymbol{X}\mathbf{\Lambda}\boldsymbol{X}^{-1} \tag{57}$$

where $\mathbf{\Lambda}$ is diagonal and consists of the eigenvalues. If the eigenvalues are negative then the covariance matrix is not positive definite. The simple solution is then to replace these values with a small positve value. However the size of this positive number might depend on the case considred. In order to get a number which is generic, we study the eigenvalues of the cyclic covariance function. These eigenvalues are found as the Fourier transform of the covariance function, which is known to be positive definite. Thus a natural selection for the smalest eigen value of the reshaped covariance matrix is the smallest eigen value of the cyclic covariance function.

We define λ_{min} as the smallest magnitude in the Fourier domain of the smoothed and upscaled covariance function

$$\lambda_{\text{min}} = \min\left(|\mathcal{F}\{\boldsymbol{C}_{us}\}|\right) \tag{58}$$

We replace all small and possibly non-positive eigenvallues of $\mathbf{\Sigma}_{us,v}$ with λ_{min}:

$$\mathbf{\Lambda}^* = \max\left(\mathbf{\Lambda}, \lambda_{\text{min}}\right) \tag{59}$$

The adjusted covariance matrix is set to be

$$\mathbf{\Sigma}^*_{us,v} = \boldsymbol{X}\mathbf{\Lambda}^*\boldsymbol{X}^{-1} \tag{60}$$

This matrix has the property of being positive definite with all eigenvalues positive. Now the matrix can be used in Bayesian inversion.

Reverting effects of positive definiteness to obtain "true" inverted covariance matrix

The reverse process of modifying eigenvalues in the eigendecomposition for obtaining a positive definite matrix, is not of interest in this context. We are rather interested in removing the effects of the adjustment on the solution of the problem, and only keeping the effects of the Bayesian inversion. Considering the inverse problem for the covariance function in the Fourier domain, we note that Bayesian inversion only affects low frequencies, and mostly effects the average level of the covariance function. Thus the largest reduction in uncertainty should be seen in the level of the parameters, i.e in the first component in the Fourier domain.

We thus define a reference value as the difference in the Fourier domain between the first element in the covariance function with adjusted eigenvalues and the first element in the posterior covariance function:

$$m_{\text{reference}} = \mathcal{F}\{\boldsymbol{C}_{us}^{*}\}(1,1,1) - \mathcal{F}\{\boldsymbol{C}_{us}^{*+}\}(1,1,1) \tag{61}$$

If the change reduction in the variance is not more than a certain precentage og this reduction then the contribution of the inversion to this component is considered to be negeligable.

Then looping through all cells, we consider the difference in the Fourier domain between the elements in the adjusted covariance matrix and the elements in the posterior covariance matrix:

$$m = \mathcal{F}\{\boldsymbol{C}_{us}^{*}\}(x_i, y_j, z_k) - \mathcal{F}\{\boldsymbol{C}_{us}^{*+}\}(x_i, y_j, z_k) \tag{62}$$

If the difference is larger than a certain limit, here set to be νm_{ref}, $\nu = 0.05$ (corresponding to a reduction larger tna 5% of the base value), we find a factor to multiply the original upscaled covariance function, to incorporate the effects of the Bayesian inversion:

$$m > \nu\, m_{\text{reference}}$$

Define the wanted ratio

$$r = \frac{\mathcal{F}\{\boldsymbol{C}_{us}^{*+}\}(x_i, y_j, z_k)}{\mathcal{F}\{\boldsymbol{C}_{us}^{*}\}(x_i, y_j, z_k)} \tag{63}$$

However, to avoid increase of variance this ratio should never be larger than 1:

$$R(x_i, y_j, z_k) = \min\left(r,\ 1\right) \tag{64}$$

R is now a matrix consiting of factors. To incorporate the effect of the Bayesian inversion we use the factors in the matrix R to multiply the upscaled *p*rior covariance function in the Fourier domain:

$$\begin{aligned} \mathcal{F}\{\boldsymbol{C}_{us}^{+}\} &= R. \cdot \mathcal{F}\{\boldsymbol{C}_{us}\} \\ \boldsymbol{C}_{us}^{+} &= \mathcal{F}^{-1}\{R. \cdot \mathcal{F}\{\boldsymbol{C}_{us}\}\} \end{aligned} \tag{65}$$

For small differences in the Fourier domain between the element in the adjusted covariance matrix and the posterior covariance matrix, we keep the element from the prior covariance matrix, hence the factor in the R matrix is set to 1. Algorithm 4 summarizes these steps.

for cell (x_i, y_j, z_k) **do**
  $m = \mathcal{F}\{\boldsymbol{C}_{us}^{*}\}(x_i, y_j, z_k) - \mathcal{F}\{\boldsymbol{C}_{us}^{*+}\}(x_i, y_j, z_k)$
  if $m > \nu m_{\text{ref}}$ **then**
    $r = \frac{\mathcal{F}\{\boldsymbol{C}_{us}^{*+}\}(x_i, y_j, z_k)}{\mathcal{F}\{\boldsymbol{C}_{us}^{*}\}(x_i, y_j, z_k)}$
    $R(x_i, y_j, z_k) = \min(r,\ 1)$
  else
    $R(x_i, y_j, z_k) = 1$
  end if
end for
$\mathcal{F}\{\boldsymbol{C}_{us}^{+}\} = R. \cdot \mathcal{F}\{\boldsymbol{C}_{us}\}$

Algorithm 4. Algorithm for finding factors to multiply the prior covariance matrix to incorporate effects of Bayesian inversion.

7 Example with synthetic data

We include a MATLAB example with synthetic data to illustrate the gravimetric inversion described in the above sections. We also want to show that the upscaling of the inversion problem does not add significant errors and can be used to overcome the possibly large computational and memory costs of the original fine scale problem.

In the example we mimic the geometric outline of observations points above Utsira formation in the Sleipner Project. We model a reservoir at the depth of 1000 m below sea bed, extending about 1 km in easting direction and about 2 km in northing direction. Observations points are placed mainly on a line in easting direction, with the majority of points outside the outline of the reservoir. The geometry is shown in Figure 9.

We discretize the reservoir with $128 \times 256 \times 64$ grid cells in east, north and vertical direction respectively. The physical dimension of one grid cell is $8\ \text{m} \times 7.5\ \text{m} \times 5\ \text{m}$. We assign a density value for each grid cell. Note that this is a total density of the subsurface, not density of for instance CO_2. Density values are generated by a Gaussian random field with mean $2\ \text{g cm}^{-3}$ and variance 0.05. The front page illustration shows the gravity response from the reservoir (outlined with dotted lines) in the observation points, marked with black circles. This computed gravity response from the reservoir is used as synthetic data for the gravimetric inversion in this example.

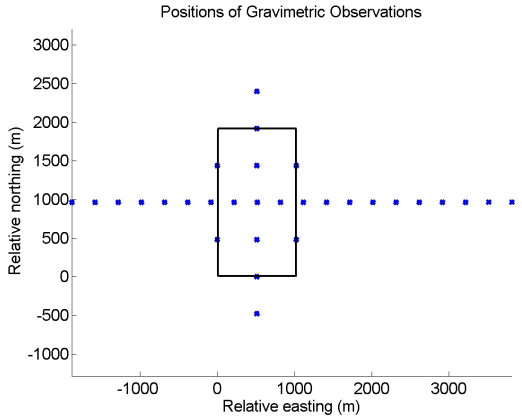


Figure 9. Observations points marked with blue relative to outline of reservoir. Note that most of the observations points are outside the region that is right above the reservoir.

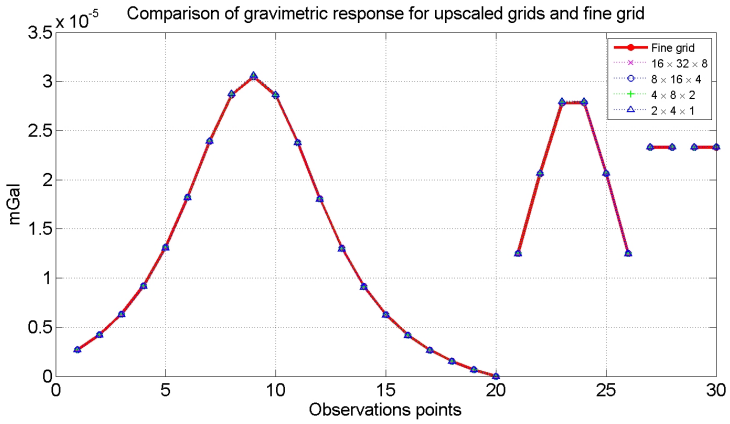


Figure 10. Plot of synthetic gravity data (red line) as well computed gravimetric response from upscaled reservoir grid models. The plot shows very small differences in gravimetric response from upscaled reservoirs, also when the upscaling factor is very high.

7.1 Upscaling of the forward model

We first use the forward model, Eq. (7), however without error term and contribution from volume outside the reservoir to compute gravity response of a set of upscaled reservoir models. We upscale the density field and the grid cell dimensions and for all observation points compute the gravity response contribution. Figure 10 shows how overlapping the gravity reponses for the upscaled models are with the fine grid gravity response (the synthetic data). This result is quite reasonable since the density field is indeed close to a constant, and the distance from the sea bed to the top of the reservoir ($1000\ m$) is relatively large, such that the approxmiation in Eqaution (32) is indeed satisfactory.

7.2 Upscaled Bayesian inversion

Recall that the gravimetric observations in themselves do not provide stand-alone data for the seismic parameters due to the unknown contribution to the response from the volume outside the reservoir. Instead, we use the difference between two gravimetric observations to obtain the difference in the seismic parameter, refer Equation (15). We therefore generate a second synthetic gravimetric respons on the fine grid, such that we can use the *difference* between the two data sets in the conditinal inverse problem. We assign the difference in gravimetric response to arise from the reservoir. This difference is plotted as slices in all three directions in Figure 11, Figure 12 and Figure 13, as the lower rightmost plot. We see that the difference is a small increase in the middle of the reservoir along the y-direction.

We upscale the reservoir to dimensions $8 \times 16 \times 4$ (upscaling factor of 16 in all directions). Prior mean is a constant field of value 0.00004. The prior covariance function has correlation range $2000,\ 500,\ 400$ in the three directions. A small error term, $\mathbf{\Sigma}_\epsilon$, is added as in Eq. (22), with diagonal elements equal to 1% of the mean of $\mathbf{G\Sigma G}^T$ in Eq. (22).

The results are plotted as slices in all three directions, for both the mean and covariance function, with the latter plotted both in the real domain and Fourier domain. For each variable we plot the prior, the smoothed prior, the upscaled prior and then the upscaled posterior, smoothed posterior and final posterior. Refer to the mnemonic scheme in Figure 2. For the covariance function in the real domain, the upscaled covariance matrix is also included in the plots.

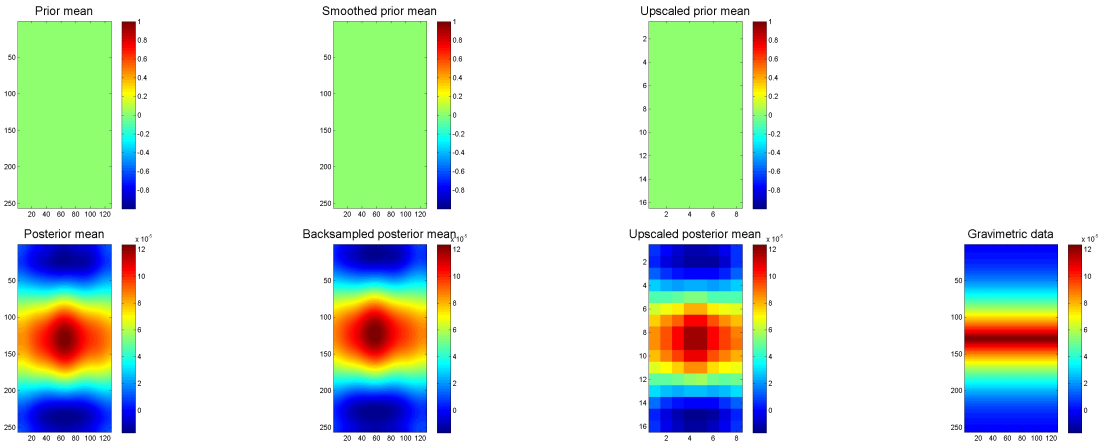


Figure 11. xy-slices of the stochastic mean. Upper row shows the upscaling of the prior mean, with the leftmost plot being on the origianl scale, the second left being the smoothed mean, while the rightmost is the upscaled mean. The lower row shows the reverse upscaling of the posterior mean. Second rightmost is the upscaled posterior, second left is the backsampled posterior mean, and the leftmost is the final posterior mean on the original scale. Lower rightmost plot shows the corresponding gravimetric data.

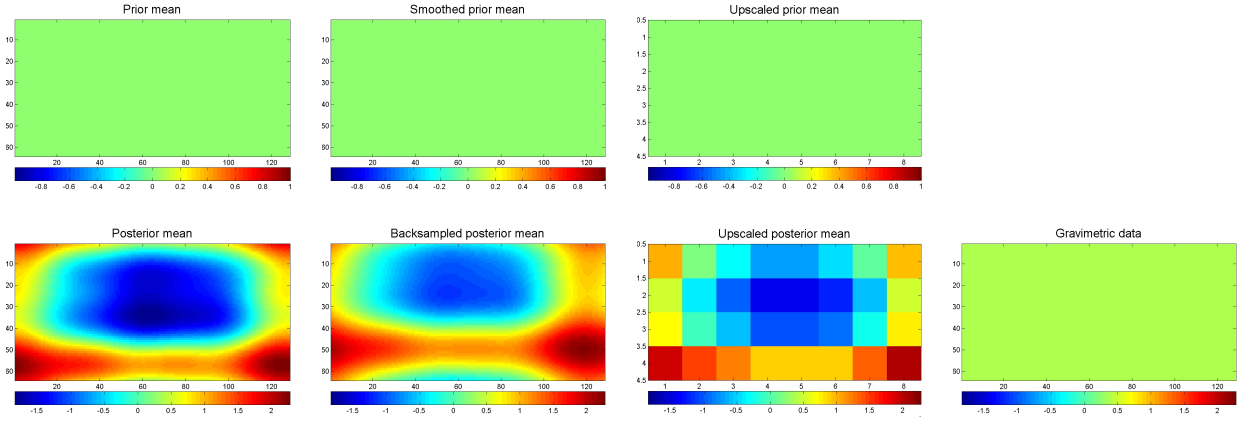


Figure 12. xz-slices of the stochastic mean. The order of the different stages of the variable is described in the caption to Figure 11.

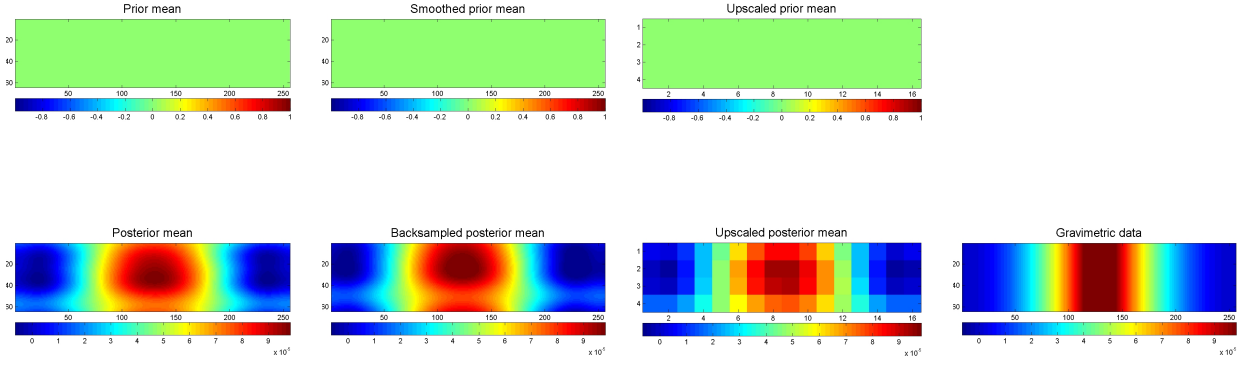


Figure 13. yz-slices of the stochastic mean. The order of the different stages of the variable is described in the caption to Figure 11.

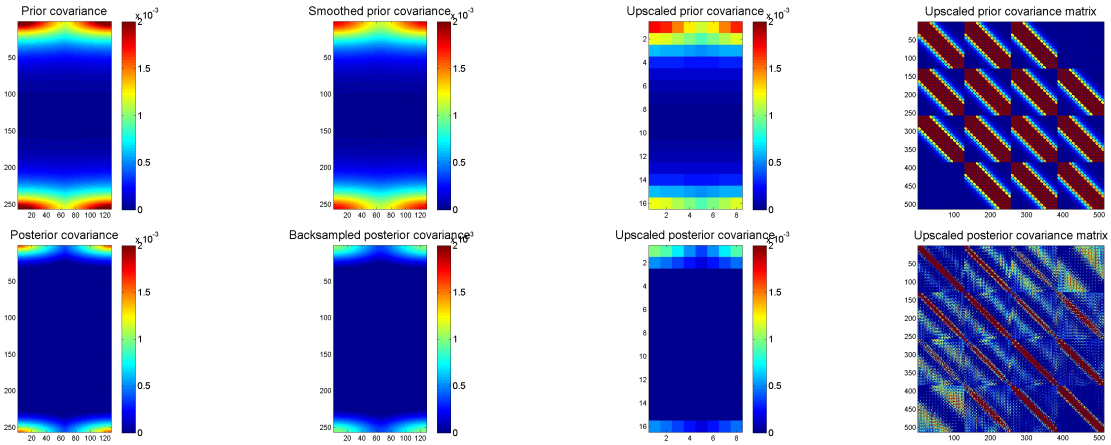


Figure 14. xy-slices of the stochastic covariance function. Upper row shows the upscaling of the prior covariance, with the leftmost plot being on the origianl scale, the second left being the smoothed covariance, while the second right is the upscaled covariance function. The rightmost plot is the upscaled prior covariance matrix. The lower row shows the reverse upscaling of the posterior covariance. The rightmost plot is the upscaled posterior covariance matrix. Second rightmost is the upscaled posterior covariance function, second left is the backsampled posterior covariance, and the leftmost is the final posterior covariance function on the original scale.

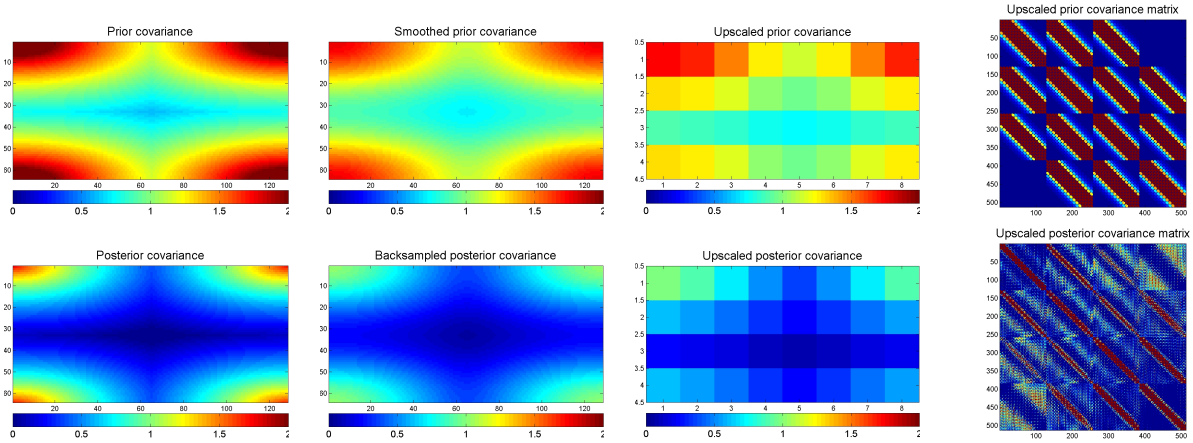


Figure 15. xz-slices of the stochastic covariance function. The order of the different stages of the variable is described in the caption to Figure 14.

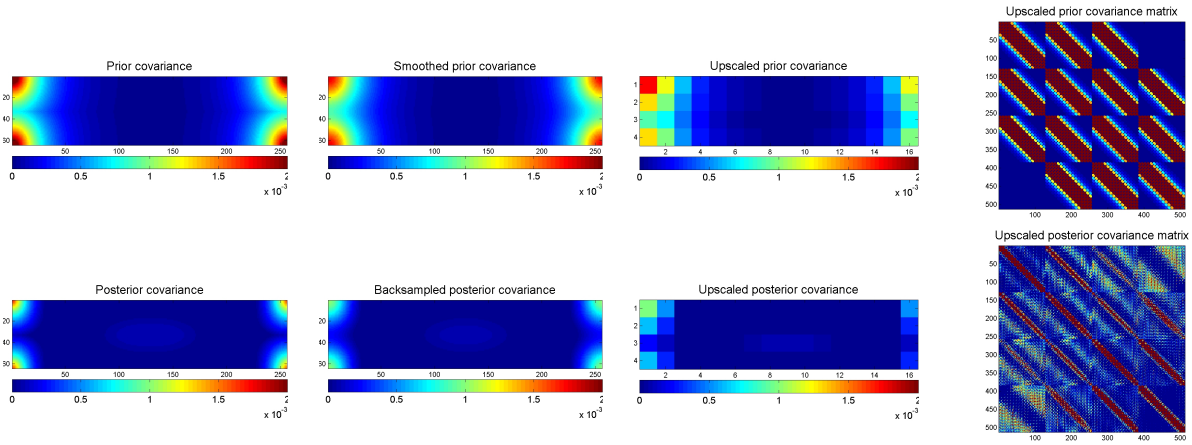


Figure 16. yz-slices of the stochastic covariance function. The order of the different stages of the variable is described in the caption to Figure 14.

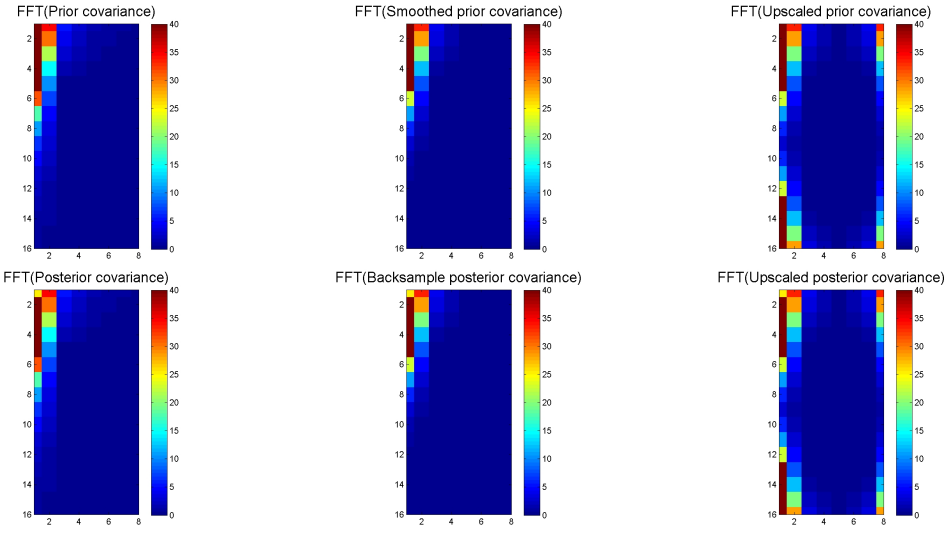


Figure 17. xy-slices of the stochastic covariance function in the frequency domain. The order of the different stages of the variable is described in the caption to Figure 14. The covariance matrix is not reported in this figure.

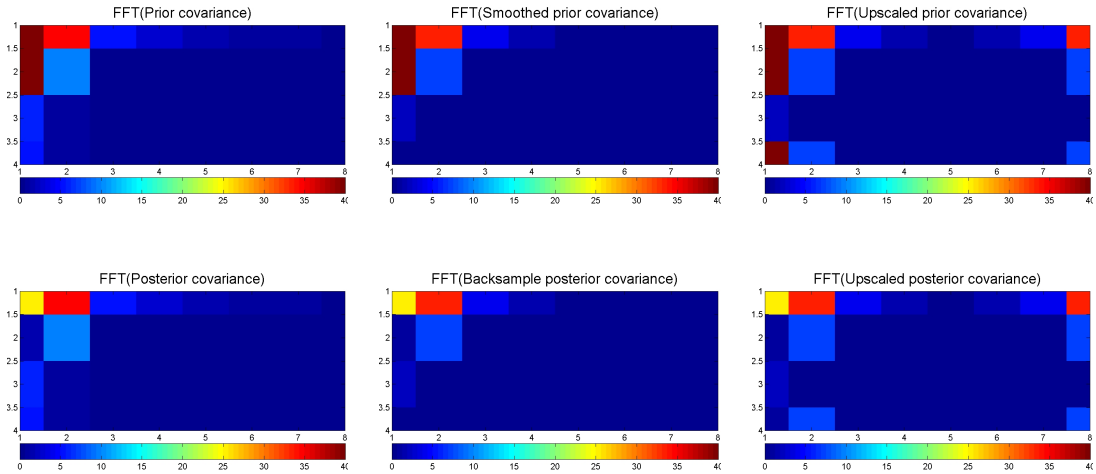


Figure 18. xz-slice of the stochastic covariance function in the frequency domain. The order of the different stages of the variable is described in the caption to Figure 14. The covariance matrix is not reported in this figure.

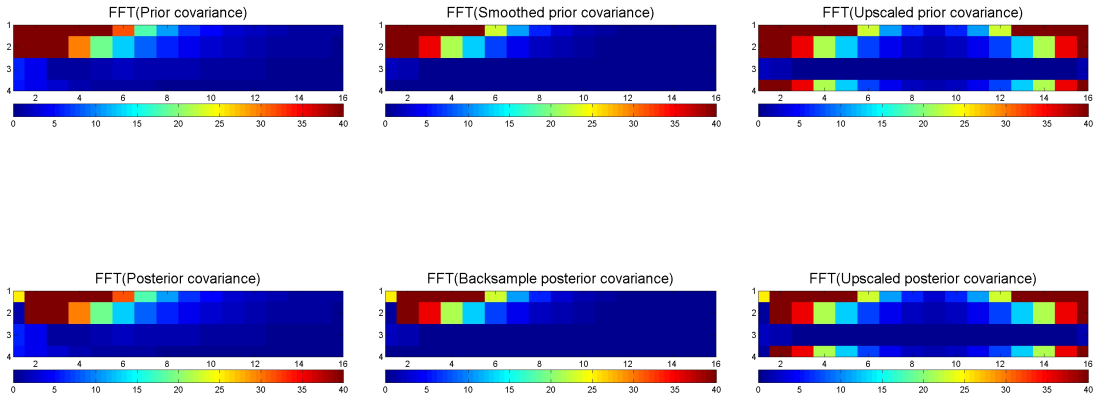


Figure 19. yz-slice of the stochastic covariance function in the frequency domain. The order of the different stages of the variable is described in the caption to Figure 14. The covariance matrix is not reported in this figure.

8 Gravimetric inversion in the 4D inversion setting

In the joint 4D seismic inversion loop, the seismic parameters are conditioned by different inversion steps in a sequential order. The inversion steps consist of the seismic inversion, travel time inversion and gravimetric inversion. These are all performed sequentially in a time loop and performed for suitable vintages for all data types. The final posterior model of the seismic parameters is the outcome of the sequence of inversions at different time steps.

The gravimetric inversion needs therefore to be incorporated in the framework of seismic amplitude and traveltime inversion. The actual realization of the framework is in CRAVA.

9 Conclusions

We document inversion of gravimetric data in a geomodel. The approach is fittet into the general framework of 4D inversion documented in separate notes. In order to overcome the computational compexity, we apply upscaling to reduce the number of unknown parameters in the inversion. We preform an geostatistical inversion on the upscaled grid, and back transforms the estimates and uncertainty to the original scale of the geomodel. The full inversion scheme is integrated in a framework of stationary observations such that the posterior uncertainty is represented with a stationary covariance function. The approach has been tested in an example, which shows that the data carry information about low-frequent components in the density change. Thus the results from gravimetric inversion alone is not suited for inferring local details about the density, but can substantiate claims to be made on a larger scale. This is in good agreement with previously published results which generally use the gravimetric data to estimate properties on a larger scale, such as the average density of CO_2 in the formation where it is injected.

References

Alnes, H., Eiken, O., Nooner, S., Sasagawa, G., Stenvold, T., and Zumberge, M. (2011). Results from sleipner gravity monitoring: Updated density and temperature distribution of the CO2 plume. *Energy Procedia*, 4(0):5504 – 5511. 10th International Conference on Greenhouse Gas Control Technologies. Available from: `http://www.sciencedirect.com/science/article/pii/S1876610211008150`.

Alnes, H., Eiken, O., and Stenvold, T. (2008). Monitoring gas production and CO2 injection at the Sleipner field using time-lapse gravimetry. *Geophysics*, 73(6):WA155–WA161.

Kolbjørnsen, O. and Kjønsberg, H. (2011). Joint 4D inversion of multiple data sources for CO2 monitoring. Technical Report SAND/20/11, Norsk Regnesentral.

Nooner, S. L., Eiken, O., Hermanrud, C., Sasaqawa, G. S., Stenvold, T., and Zumberge, M. A. (2007). Constraints on the in situ density of CO2 within the Utsira formation from time-lapse seafloor gravity measurements. *International Journal of Greenhouse Gas Control*, 1(2):198–214.

A Lognormality of the seismic parameters

The seismic parameters given in $\boldsymbol{m}$ follow a lognormal distribution. That means that it is the logarithm of the variables that follows a normal distribution:

$$\log \boldsymbol{m} = [\log \boldsymbol{v}_p, \log \boldsymbol{v}_s, \log \boldsymbol{\rho}] \sim \mathcal{N}(\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m) \tag{A.1}$$

For small standard deviations compared to the expectations for all components in the multinormal distribution, we can assume for a lognormal distribution

$$\log [\boldsymbol{m} \sim \mathcal{N}(\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)] \tag{A.2}$$

that also the parameter itself is approximately normally distributed

$$\boldsymbol{m} \sim \mathcal{N}(\boldsymbol{\mu}_m^*, \boldsymbol{\Sigma}_m^*) \tag{A.3}$$

with the following transformations between the expectations and variances:

$$\begin{aligned} \boldsymbol{\mu}_m^* &= \exp^{\boldsymbol{\mu}+0.5\sigma^2}, \quad \sigma^2 = \text{diag}(\boldsymbol{\Sigma}_m) \\ \boldsymbol{\Sigma}_m^* &= (\exp^{\boldsymbol{\Sigma}_m} - 1). \cdot \boldsymbol{\mu}^* (\boldsymbol{\mu}^*)^T \end{aligned} \tag{A.4}$$

In the explanations in this note, we write all expressions using only the seismic parameters $\boldsymbol{m}$ and its normal distribution. However, in the final calculations in the CRAVA software, which is where the gravimetric inversion eventually will be implemented, the inversion will be done using the logarithm of the seismic parameters.

B MATLAB example of one-dimensional upscaling

For the sake of intuitive visualization and verification of the upscaling steps, we include a simple one-dimensional example with complete MATLAB code. Following what happends in the smoothing and subsampling for a one-dimensional line is easier than for a multi-dimensional example, and thus documenting the example has been a good source of learning.

We have a original fine grid of 128 cells and want to upscale the grid to have 8 coarse grid blocks. Each coarse grid block thus consists of `nb = 16` cells.

```
nx    = 128; % original grid
nx_up = 8;   % Upscaled grid
nb    = nx/nx_up;
```

We have a one dimensional random variable `A`, plotted with blue color in Figure B.1. Next, the convolving kernel is defined as an averaging function

```
h          = zeros(1,nx);
h(1, 1:nb) = 1/nb;
```

We Fourier transform the random variable and the convolution kernel

```
fft_A = fftn(A);
fft_h = fftn(h);
```

Applying the convolution theorem lets us do convolution as point wise multiplication in the Fourier domain. Note that we use the complex conjugate of the convolution kernel.

```
fft_A_s  = (fft_A).*conj(fft_h);
```

To find the representation of the smoothed random variable, we take the inverse Fourier transform of the smoothed random variable. Note that we expect the variable to only have real numbers, hence we take the real part of the inverse Fourier transform. The smoothed variable is plotted with green color in Figure B.1.

```
A_s  = real(ifftn(fft_A_s));
```

Subsampling in the Fourier domain corresponds to keeping the low frequencies. Here we consider the first `nx_up/2` frequencies as low frequencies as well as the last `nx_up/2-1` frequencies in the MATLAB representation

```
fft_A_us = fft_A_s([1:nx_up/2, (nx-(nx_up/2)+1):nx]);
```

We take the inverse Fourier transform of the subsampled variable. Again we only keep the real component of the transform.

```
A_us = real(ifftn(fft_A_us));
```

In the process above, we have done two forward FFT operations and one inverse FFT. In addition, the averaging convolution kernel introduces scaling into the expression. We therefore need to re-scale the inverse Fourier tranform to get comparable values to plot. In Matlab the scaling is `1/nb`. This variable is plotted in red color in Figure B.1.

```
A_us = A_us*1/nb;
```

To exemply both strategies for subsampling, we now subsample in the real domain. We select every cell in the smoothed random variable that corresponds to the first cell in each coarse grid block to obtain the subsampled variable. The black line in Figure B.1 shows this variable.

```
A_us2     = A_s([1:nb:nx]);
```

In Figure B.1 note the shift in plotting of the smoothed values. Even though smoothing gives one value for each of the original values, then the first value does indeed correspond to the first cell in the coarse grid block. The first cell in the coarse grid block is best represented by the position of (approximately) the mid-cell of the coarse grid block. We therefore shift the smoothed variable to match up this representation. Also note the stair-plotting of the upscaled values.

```
figure;
plot(1:nx, A, ...
     nx/nx_up/2+1:(nx+nx/nx_up/2), A_s, 'g-', 'LineWidth', 2);
hold on;
stairs(1:nx/nx_up:(nx+nx/nx_up), [A_us  A_us(end)],  'r-', 'LineWidth', 2);
stairs(1:nx/nx_up:(nx+nx/nx_up), [A_us2 A_us2(end)], 'k-', 'LineWidth', 2);
legend('Original A', 'Smoothed A_s', ...
       'Upscaled in FFT domain: A_{us}', ...
       'Upscaled in real domain: A_{us2}', ...
       'Location', 'Best');
xlabel('x'); ylabel('A(x)');
```

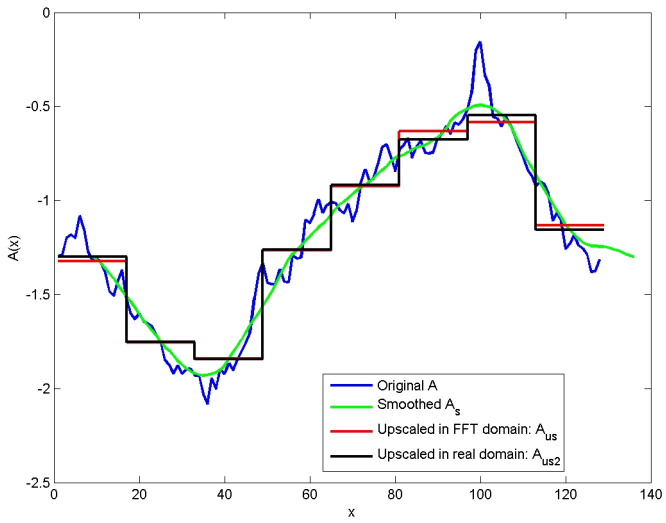


Figure B.1. One dimensional example: Blue line is the original variable, the green line is the smoothed variable, the red line is the upscaled variable with subsampling in the Fourier domain, and lastly, the black line is the upscaled variable with subsampling in the real domain.

We now "do something" in the upscaled domain. In this case we add a constant to the variable. The changed variable is plotted as the red line in Figure B.2. Note the scale of the y-axis compared to Figure B.1.

```
B_us = A_us + 10;
```

Firstly, in the reverse process of upscaling, we Fourier transform the upscaled and modified random variable. Note that we reverse the scaling introduced of the inverse Fourier transformed upscaled variable.

```
fft_B_us = fftn(B_us*nb);   % Scale before doing fft
```

Backsampling in the Fourier domain consists of putting back modified low frequencies in the smoothed (unchanged) variable. The backsampled variable is plotted as the green line in Figure B.2.

```
fft_B_s = fft_A_s;
fft_B_s([1:nx_up/2, (nx-(nx_up/2)+1):nx]) = fft_B_us;
B_s = real(ifftn(fft_B_s));
```

We carry out pseudo deconvolution. Note the special treatment where the Fourier transform of the convolving kernel is 0. For those frequencies we keep the original values from Fourier transform of the original variable. Note also the use of the complex conjugate of the Fourier tranform of the convolution kernel.

```
fft_B = fft_B_s./conj(fft_h);
I = find(fft_h == 0);
fft_B(I) = fft_A(I);
```

Lastly, we take the inverse Fourier tranform of the backsampled and deconvolved variable to find the fine-scale representation of the changed variable. Note no scaling.

```
B  = real(ifftn(fft_B));
```

Plotting of the changed and down-scaled variable is shown in Figure B.2 as the blue line. Note the identical shape of the line compared to the original variable in Figure B.1.

```
figure;
stairs(1:nx/nx_up:(nx+nx/nx_up), [B_us  B_us(end)], 'r-', 'LineWidth', 2);
hold on;
plot(nx/nx_up/2+1:(nx+nx/nx_up/2), B_s, 'g-', ...
     1:nx, B, 'b-', 'LineWidth', 2);
legend('Upscaled: B_{us}', ...
       'Backsampling in FFT domain: B_s', ...
       'Deconvolved: B',...
       'Location', 'Best');
xlabel('x'); ylabel('A(x)');
```

To verify the process, we subtract the original random variable from the changed random variable. The desired result is that the difference is the constant added to the upscaled variable.

```
mean(B-A)
```

```
ans =

   10.0000
```

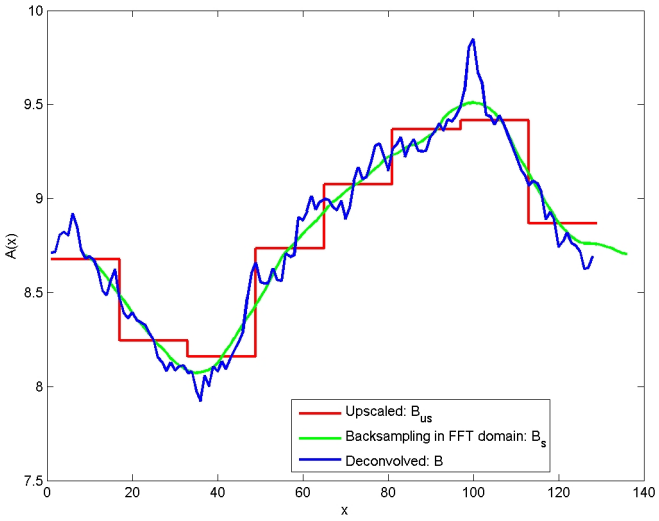


Figure B.2. The 1D example after a change has been done to the variable. The red line is the changed upscaled variable, the green line is the variable that is backsampled in the Fourier domain, and finally, the blue line is the deconvolved fine-scale variable. Note the identical shape of the blue line compared to Figure B.1 and the shift on the y-axis.

C MATLAB functions

C.1 Find lag index

```
function lag_index_grid = MakeLagIndexGrid(nx_up, ny_up, nz_up)

lag_index_grid = zeros(nx_up*ny_up*nz_up, nx_up*ny_up*nz_up, 3);

for k1 = 1:nz_up,
    for j1 = 1:ny_up,
        for i1 = 1:nx_up,
            I =  i1 + (j1-1)*nx_up + (k1-1)*nx_up*ny_up;
            for k2 = 1:nz_up,
                for j2 = 1:ny_up,
                    for i2 = 1:nx_up,
                        J = i2 + (j2-1)*nx_up + (k2-1)*nx_up*ny_up;
                        lag = [i2, j2, k2] - [i1, j1, k1];
                        if(abs(lag(1)) <= nx_up/2 && abs(lag(2)) <= ny_up/2 && abs(lag(3)) <= nz_up/2)
                            if(lag(1) >= 0)
                                ind1 = lag(1) + 1;
                            else
                                ind1 = nx_up + lag(1) + 1;
                            end
                            if(lag(2) >= 0)
                                ind2 = lag(2) + 1;
                            else
                                ind2 = ny_up + lag(2) + 1;
                            end
                            if(lag(3) >= 0)
                                ind3 = lag(3) + 1;
                            else
                                ind3 = nz_up + lag(3) + 1;
                            end
                            lag_index_grid(I,J,:) = [ind1, ind2, ind3];
                        else
                            lag_index_grid(I,J,:) = [-1,-1,-1];
                        end
                    end
                end
            end
```

```
        end
    end
end
```

C.2 Reshape according to lag

```
function A = ReshapeAccordingToLag(A_3D, nx_up, ny_up, nz_up, lag_index)

A = zeros(nx_up*ny_up*nz_up, nx_up*ny_up*nz_up);
for k1 = 1:nz_up,
    for j1 = 1:ny_up,
        for i1 = 1:nx_up,
            I =  i1 + (j1-1)*nx_up + (k1-1)*nx_up*ny_up;
            for k2 = 1:nz_up,
                for j2 = 1:ny_up,
                    for i2 = 1:nx_up,
                        J = i2 + (j2-1)*nx_up + (k2-1)*nx_up*ny_up;
                        if(lag_index(I,J,1)==-1 && lag_index(I,J,2)==-1 && lag_index(I,J,3)==-1)
                            A(I,J) = 0;
                        else
                            A(I,J) = A_3D(lag_index(I,J,1),lag_index(I,J,2),lag_index(I,J,3));
                        end
                    end
                end
            end
        end
    end
end

return;
```

C.3 Reshape back to 3D

```
function A_3D = ReshapeBackTo3D(A, nx_up, ny_up, nz_up, lag_index)

A_3D  = zeros(nx_up, ny_up, nz_up);
sum_A = zeros(nx_up, ny_up, nz_up);
count = zeros(nx_up, ny_up, nz_up);

for k1 = 1:nz_up,
    for j1 = 1:ny_up,
        for i1 = 1:nx_up,
            I =  i1 + (j1-1)*nx_up + (k1-1)*nx_up*ny_up;
            for k2 = 1:nz_up,
                for j2 = 1:ny_up,
                    for i2 = 1:nx_up,
                        J = i2 + (j2-1)*nx_up + (k2-1)*nx_up*ny_up;
                        if(lag_index(I,J,1) > 0 && lag_index(I,J,2) > 0 && lag_index(I,J,3) > 0)
                            i = lag_index(I,J,1);
                            j = lag_index(I,J,2);
                            k = lag_index(I,J,3);
                            sum_A(i,j,k) = sum_A(i,j,k) + A(I,J);
                            count(i,j,k) = count(i,j,k) + 1;
                        end
                    end
                end
            end
        end
    end
end

for k1 = 1:nz_up,
    for j1 = 1:ny_up,
        for i1 = 1:nx_up,
            if(count(i1,j1,k1)>0)
                A_3D(i1,j1,k1) = sum_A(i1,j1,k1)/count(i1,j1,k1);
            end
        end
    end
end
```